Iron Bank: Site Reliability Engineering

Abstract: The Iron Bank can be defined as the service composed of the web facing front sites Repo1(dccscr.dsop.io), DCAR(dcar.dsop.io) and the VAT(vulnerability assessment tracker: soon to be launched) and the back end services that populate these front websites. From the inception of the idea, the Iron Bank has been targeted not only to internal DOD users, but also to the wider community across the internet. If we are going to attempt to drive wide adoption of the Iron Bank it is imperative that it is perceived as a reliable web resource. The CHT (Container Hardening Team) and Platform1 teams as a whole should implement procedures and agreements with the purpose of encouraging site wide reliability and reducing the time to respond in the case of a reliability failure.

> *"Software engineering has this in common with having children: the labor before the birth is painful and difficult, but the labor after the birth is where you actually spend most of your effort."*
>
> *- Site Reliability Engineering, by Niall Richard MurphyBetsy BeyerChris Jones-Jennifer Petoff*

Important Concepts

When following the concepts of DevOps as applied to a specific technical project, it becomes clear that the availability of the project should be just as important to the developers as the features and security. Developers should look at the project as a whole, instead of focusing in on their component parts. This promotes a comprehensive view of the application stack and negates the tendency to blame failures on a part of the stack that a particular group or person is not responsible for. Working with complex distributed systems requires troubleshooting the whole stack.

Emergency Response

Our goal in monitoring the application stack it to provide stability and uptime to the end user while still allowing for the continual application of patches and feature releases. The best metric available to correlate with our goal is availability. Availability is a formula of Mean Time Between Failures (MTBF) and Mean Time To Respond (MTTR)

$$A(MTBF, MTTR) = \frac{MTBF}{(MTBF + MTTR)}$$

Infrastructure

The infrastructure for the Iron Bank is provided as Infrastructure as Code from the Platform1 team deployed onto Cloud 1 Development Environment. The specific nature of our Infrastructure (being Kubernetes deployed as IaC) means that there should not be a significant amount of support needed to maintain this layer of the service. The Platform1 team should provide (in the IaC) a method of Authenticating and providing role based access to necessary services.

Change Management

This should mostly be handle by GitOps. All Changes and Deployments should be to code which is then tested and deployed from git through a CI/CD pipeline.

> *"By removing humans from the loop, these practices avoid the normal problems of fatigue, familiarity/contempt, and inattention to highly repetitive tasks. As a result, both release velocity and safety increase."*
>
> *Site Reliability Engineering*

Capacity Planning, Provisioning, Efficiency and Performance

All of these tasks are interrelated. Capacity planning is the method by which we ensure that there is sufficient capacity and redundancy to server projected demand. Provisioning is directly related to capacity planning; if the projections show that capacity is too great or too small then system provisioning should happen immediately. Excess capacity is expensive, and lack of capacity negatively affects reliability which leads to correct resource usage. Efficiency, while seemingly focused on cost, is directly related to reliability as mentioned previously.

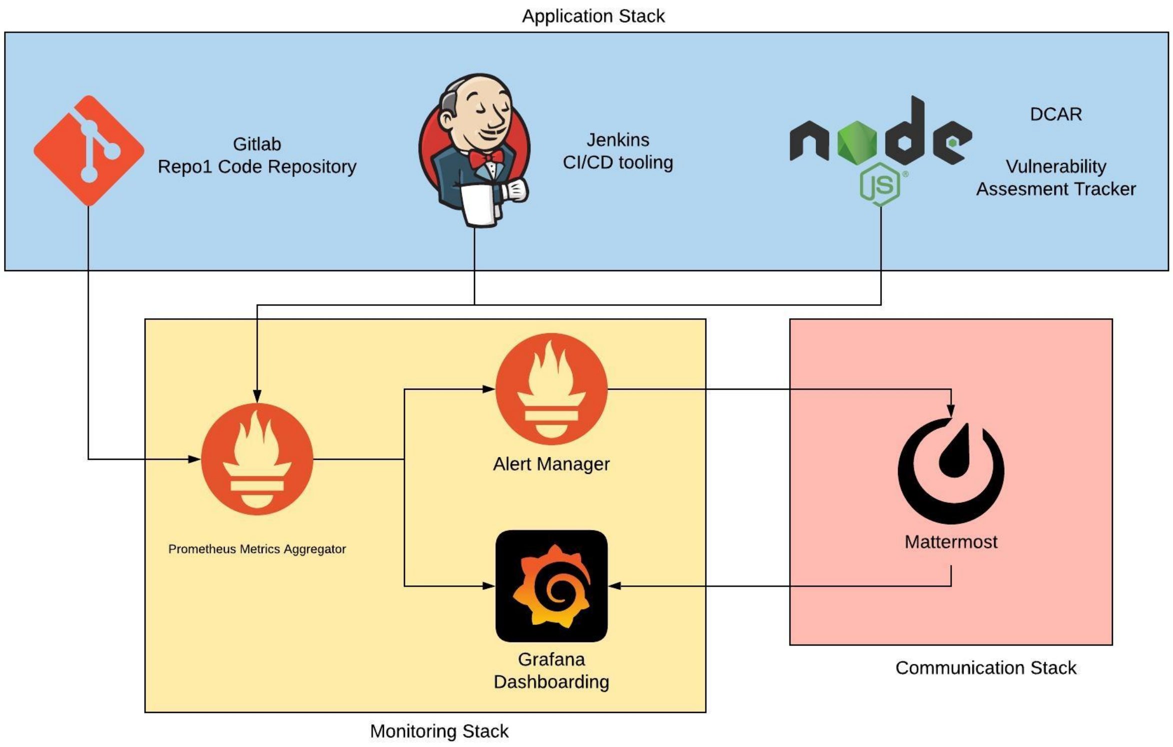
Monitoring Tools

Included in the Platform One stack are a bevy of monitoring tools. These tools are either free or covered by the subscriptions for the platform. Prometheus handles metrics collection and calculation, while Grafana is a powerful dashboarding tool. Alert Manager is a flexible alerting tool that can be setup to work with almost any communication tool. Mattermost is the chosen communication tool for the platform so I suggest pointing all alerts to a Mattermost channel created for that specific part of the application stack. Using this strategy allows free flow of communication, ease of creating and updating both monitoring and alerting rules, and easy control of access to alerts.

By having developers and operators of the applications stack leverage the monitoring and alerting tools available in the Platform One infrastructure we can gather important metrics and correlate key performance indicators that will allow us to not only track the availability of our application but also plan for patches and feature releases, and right sizing of the infrastructure components.

(Diagram attached)

Iron Bank Suggested Monitoring Strategy


Application Stack
Gitlab
Repo1 Code Repository
Jenkins
CI/CD tooling
DCAR
Vulnerability
Assesment Tracker
Prometheus Metrics Aggregator
Alert Manager
Grafana
Dashboarding
Mattermost
Communication Stack
Monitoring Stack